Remarks

In the application, claims 1 through 7, 16, 17, and 22 through 24 are pending. No claims currently stand allowed.

The Final Office Action dated May 17, 2004, has been carefully considered. The Final Office Action rejects claims 1 through 10 and 12 through 24 under 35 U.S.C. § 102(b) as anticipated by U.S. Patent 5,790,861 ("Rose"). Claim 11 is rejected under 35 U.S.C. § 103(a) as obvious in light of Rose and U.S. Patent 6,385,769 ("Lewallen").

The present application and Rose both describe techniques for replacing a base software object class with another object class. However, the timing of this replacement differs, and that difference significantly affects the utility of these techniques.

Rose describes traditional base-class inheritance where the replacement *always occurs at compile or link time*. Applicants respectfully disagree that the portion of Rose cited by the Final Office Action shows replacement occurring *during program execution*. In Rose's Figure 2, the base and replacement objects are compiled (step 210) and then linked (step 260). Rose's execution occurs in step 280 ("run") and *that step ends before class replacement can occur* (by looping back to steps 210, 220, and 260). Thus, in Rose the class replacement occurs *between* a first program execution and a second execution and *not during* program execution. The entire text of Rose's specification supports this view. For specific statements, see, e.g., the Abstract and column 3, lines 43 through 48 ("As a result, even if header file class definitions are subsequently modified, it will often be possible to generate updated executable code *by recompiling the modified header files and relinking the resulting object code*, without necessarily recompiling the application program code.") (emphasis added).

In contrast to Rose's compile- (or link-) time replacement, the present invention allows class replacement *during program execution*. This is true polymorphism and even works for a replacement class *that was unknown at compile time*, a feature clearly not disclosed by Rose. The present specification emphasizes this timing on page 14, lines 8 through 10 ("In contrast [with a Rose-type system], in the example given in FIG. 2, the replacement class B *may be designed after the reusable module 70 was generated.*"), page 14, lines 14 through 17 ("The class replacement in accordance with the invention enables *dynamic creation of objects of a new derived class* instead of objects of a base class *during execution of the existing code in the program.*"), page 15, line 24,